PRISENIT – A Probabilistic Search Recommendation Algorithm to Improve Search Efficiency for Network Intelligence and Troubleshooting

Mikel Zuzuarregui[*#], Enda Fallon[#], MingXue Wang[*], John Keeney[*], Paul Jacob[#]

[*] Network Management Laboratory
Ericsson Software Campus
Athlone, Ireland
{mikel.zuzuarregui, mingxue.wang, john.keeney}@ericsson.com

[#] Software Research Institute
Athlone Institute of Technology
Athlone, Ireland
{efallon, pjacob}@ait.ie

Abstract— **When searching for data in a telecommunications network management application, large search result sets are common. In order to refine the results to retrieve useful information existing systems normally require additional user intervention such as appending or removing a search keyword, adding a filter, grouping results, etc.. This work proposes a Probabilistic Search Recommendation Algorithm to Improve Search Efficiency for Network Intelligence and Troubleshooting (PRISENIT). PRISENIT is a query-based recommendation algorithm intended to improve search efficiency and usability for telecommunication system management. PRISENIT is an extension of an item-based collaborative filtering algorithm. It uses correlation-based similarity and users' implicit feedback in order to improve search efficiency. It learns from previous experiences in order to optimize decision-making. Currently there exists no known query-based recommender adaptation mechanism for network management. Existing search engines use previous user searches to make a suggestion based on the keyword. PRISENIT not only considers search terms, it also considers the influence of filters and features in order to makes network searches more efficient as it removes the necessity for users to manually choose search features or search filters. Experimental results show that PRISENIT can improve user experience in a telecommunications management environment.**

Keywords - recommender system; item-based collaborative filtering; query-based recommender; network management

I. INTRODUCTION

With the ever increasing volume of telecommunications data (end user data, performance metrics, and alarms), big data and information retrieval (search) technologies have gained attention by telecom equipment vendors [1][2]. In network management systems traditional SQL-based data-retrieval techniques are mainly used. Such systems however are performance limited for unstructured telecom data such as network node logs, alarm text and trouble ticket messages. Increasingly enterprise search technologies are capable of handling both structured data from corporate databases and unstructured data sources and have potential for management analysis. The characteristics of telecommunications network data however, volume, complexity and differing lifespans create challenges when developing network analysis solutions.

This work particularly focuses on how a search interface can be improved to assist network troubleshooting. PRISENIT can learn from previous experiences and interact with human decision techniques. In the case of system error, it can implement autonomic system reconfiguration. Unlike existing search approaches, PRISENIT considers not only query terms (keywords) but also search filters and presentation features. In other commercial solutions it is possible for users to manually store and maintain interaction templates. Within the PRISENIT approach common searches are automatically learned and recommended. PRISENIT uses correlation-based similarity and combines implicit feedback from the user in order to improve search efficiency for network intelligence and troubleshooting. Such learning and query suggestion is not currently available for telecommunication or network management domains. Data Analytics approaches (e.g. OLAP tools) also support the user to define custom search-based reports (custom filters and features to present results). However, these reports are not automatically learned, tuned or recommended to the user. As the searched data changes, or preferred ways of interacting with the system changes, our system learns and recommends new queries (terms, filters and features). PRISENIT speeds up network management system interactions for common or routine management tasks, and also learns best practices for troubleshooting, reducing skill requirements and cost in solving management issues.

In summary, the key problem is that users usually require multiple additional iterative search actions or steps after an initial query, such as appending or removing a keyword, adding a filter to find results, and then selecting presentation features to appropriately present the results. This makes search-based management complex to use and tedious and time consuming for users, while also expending system resources performing searches that are unnecessary.

We empirically evaluate PRISENIT with data from experimental evaluations undertaken at the Ericsson research lab. The system was evaluated using Ericsson network management engineers. Results illustrate the enhanced search interactivity of the PRISENIT approach.

The remainder of this paper is organized as follows. Section II will introduce related work about how to personalize search engine interfaces. Section III shows the architecture and methodology of the recommender system. Section IV will show the system application. In section V, we discuss the usability of the recommender system, and point out how a prototype implementation was developed and operated, before concluding in section VI.

II. RELATED WORK

A. Search engine

The explosive growth of Web search services has triggered an increasing demand for web personalization systems in order improve end user Web browsing experience. In the recent years such systems have become ubiquitous. Typically such systems take advantage of data about the users' past behaviors or usage. Web personalization can typically be categorized in three categories: Web Utilization Miner (WUM), user profiling and recommender systems.

Web mining techniques have been analyzed for several years. In [3] the authors investigate the importance of a suitable data set. They introduce a novel model for improving web-mining using semantic web and synaptic web. In [4] the impact of semantic web to improve web mining experience is examined by presenting a weight estimation process with span time, request count and access sequence details.

A number of approaches have been investigated in the area of user profiling. In [5] the authors provide a framework and findings in mining web-usage/navigation information from website log files, involving user profiles and external data describing the ontology of the web content. In [6] the authors propose description-logic based semantic user modelling for spatial web personalization. Recently, ontologies have been applied to obtain user profiles in [7].

B. Web personalization

The challenge of Web personalization and usability can be addressed by recommender system approaches. Recommendation systems use information filtering to assist a user in a decision-making process by giving some suggestion that may benefit the user based on previous experience of that user and similar users. The suggestion could be about products, information or services by learning user preferences. A number of studies focus on recommendation systems and web personalization, e.g. using ontologies [8] or sequential web access patterns [9].

C. Recommender system

Recommendation systems first appeared circa 1997 [10]. In general, all the recommender systems are based on the analysis of item attributes, user attributes or user behavior. A recommendation system must predict the utility of a candidate item in comparison to alternatives.

Collaborating Filtering was proposed by Rich [11] and has been applied in many commercial web sites. It consists of *n* users and *m* items connected by user ratings of items. Ratings can be explicit (e.g. based on feedback) or implicit (e.g. based on uptake). Implicit feedback is more difficult to determine however it can usually provide more information and context than explicit feedback.

In [12] a query language for network search is introduced. The search space is represented with objects as a set of attribute-value pairs, additionally showing that the method allows distributed execution.

Previously in [13] we investigated the applicability of recommender systems to assist Network Operations Centre operators for network management systems. In this paper we create a monitoring and trouble-shooting system for network managers.

PRISENIT is unique in that it focuses on using a search engine for network management to improve network intelligence and troubleshooting by recommending improved query-based searches. It shows the user how to change the search parameters to reach a possible better solution, as well helping to efficiently show more meaningful information for particular use cases.

III. PRISENIT ALGORITHM

A. Transaction log

A transaction log is a log file of the communications between a system and the user of that system. It has two main parts: a) Search query: A search query part represents the complete user search requirements; b) Rating: the rating part is the feedback of result of the search

The query part is a vector $\vec{V} = [v_t, v_l, v_f]$ that can be further divided into three components; terms, filters, and features. These three parts cover the user requirement of different aspects of a search.

v_t is a user query which is composed of a number of keywords or terms. Terms are a series of characters that the users type in the search box within a query separated by space.

v_l is a set of filters applied on the search results such as a specified time range or restrictions on the data source, etc.

v_f describes presentation features with different settings to describe how the result is presented. For example, the result is sorted by time or by ranking score, the result is faceted by a node or/and severity, etc.

The rating part r is a user rating of the results returned from the query part. It is measured by user interaction / behavior with result (i.e. time spent interpreting/using the results, using scrolling, click-through, and so on).

B. The recommendation calculation

The search recommendation is based on past user search experience. PRISENIT calculates a recommendation score for records in the translation log for a target user input query. Fig. 1 illustrates the mechanism utilized by PRISENIT when calculating recommendation scores.

Within PRISENIT the computation of similarity of a query is composed of the amalgamation of the similarly of each component of the query using a pluggable correlation mechanism. Taking the term component v_t as an example, the computation of similarity value of terms of a target query with terms of a comparing query is undertaken using the Pearson correlation coefficient sim_v_t . Similarly, we compute the similarity between the filter and feature element using the similar sim_v_l and sim_v_f functions. In contrast, a set of common filters or common features is used for the measure. The selection of the Pearson correlation within PRISENIT is based on [14][15].

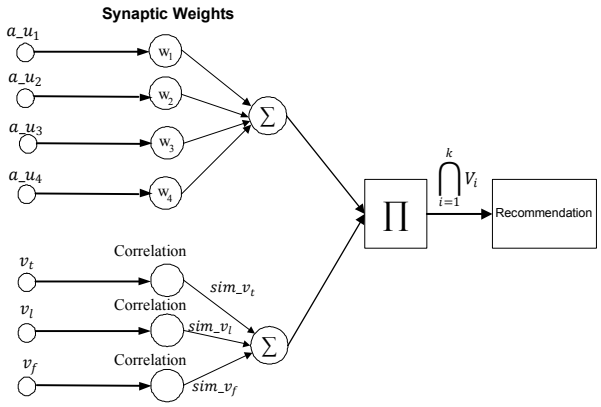

Figure 1. PRISENIT Architecture

The search rating is based on the end user interaction with the suggested results. A user action a is a measure of some type of user interaction; mouse-overs, click-throughs, viewing different results pages, scrolling windows and time spent on the page, etc.

$$r \quad \sum_{i=1}^{n} a\_u_i w_i. \tag{1}$$

w_i is a weight factor applied to each type of user interaction, $\sum_{i=1}^{n} = 1$. A higher weight is applied to user actions which apply directly to results; clicking on results or comparing actions such as time spent on the page.

In order to consider user feedback on search recommendations, a score is calculated based on both the similarity of the query and the rating.

$$s \quad sim_v * r \ . \tag{1}$$

Finally, the queries with the top recommendation scores are used to create the search recommendation. The recommended search is a combination of each component of the search query. It considers all terms, filters and features in order to create a new query.

$$V_{rec} \quad \cup_{i=1}^{k} V_i. \tag{1}$$

The union is over the k queries with the highest scores.

Table 1 illustrates the queries that are stored in the database. PRISENIT uses those queries as the basis for comparison. Table 2 illustrates the operation of the PRISENIT recommendation algorithm using the queries from Table 1. In this example, the queries with the most similarity are 2, 3 and 5.

Algorithm 1 Recommendation calculation

Input: Q String with the query
Output: String with the recommendation
Initialization finalScores;
1: split the query into terms, filters and features;
2: queries = retrieveSetsOfQueriesWithRatings();
3: **for each** queries **do**
4: sim_1_{i1} = Correlation(terms , $queries_{i,terms}$);
5: sim_1_{i2} = Correlation(filters , $queries_{i,filters}$);
6: sim_1_{i3} = Correlation(features , $queries_{i,features}$);
7: $sim_i = sim\_1_{i1} + sim\_1_{i2} + sim\_1_{i3}$;
8: normalizedPerformanceMetricsRatings(timeSpent, scrolling, click, numClicks);
9: r_i = (a_u_1 × wTime) + (a_u_2 × wClick) + (a_u_3 × wScrolling) + (a_u_4 × wNumClicks);
10: finalScores = $sim_i \times r_i$;
11: **return** buildRecommendation(finalScores, k);

Method: buildRecommendation(finalScores, k);
1: retrieve from database k queries with the highest scores, split into terms, filters and features
2: create a string with unique terms
3: create a string of unique filters taking care about their compatibility
4: create a string with unique features
5: join terms, filters and features into one string to create a query
6: return query

C. Architecture

In this work PRISENIT is designed based on Apache Solr[1]. Management data files are constantly retrieved from a managed network and then indexed for searching. The data includes fault data (FM), configuration actions (CM), and performance data (PM). Data files are retrieved from the Hadoop[2] RDBMSs and XML data stores used in the Ericsson system. Fig. 2 illustrates the architecture of the PRISENIT recommendation system within the overall architecture.

The main feature of this kind of system is the ability to predict the additional search actions or steps that may be required by users to extract required information. As a result, all useful additional steps that may be required by a user are suggested and can be applied in a single action. PRISENIT learns previous user search interactions and experiences to give suggestions for new search queries.

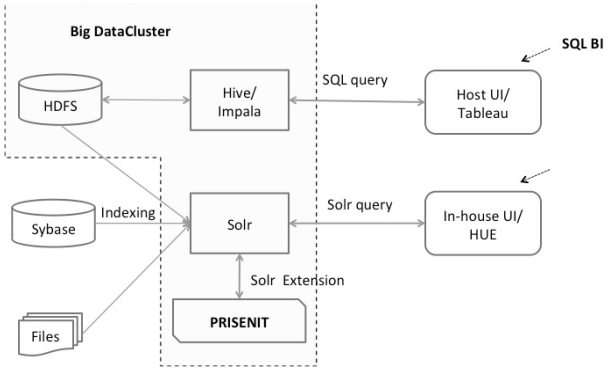

Figure 2. Overall System Architecture Incorporating PRISENIT

[1] http://lucene.apache.org/solr/
[2] http://hadoop.apache.org/

Query to Solr:
Term: failure OR Conditioner
Filters: wt=json & rows=50 & start=0 & facet.field=RNC & facet.field=data_source & hl.fl=* & hl.snippets=3 & fl=RNC,index_id,RBS & facet.field=Object_of_reference & facet.field=additional_info
Features: facet=true & hl=true

Recommendation:
Term: failure OR rnc80 OR rnc95
Filters: wt=json & rows=50 & start=0 & facet.field=RNC & facet.field=data_source & facet.field=Problem_text & hl.fl=* & hl.snippets=3 & fl=RNC,index_id,RBS & facet.field=Object_of_reference & facet.field=additional_info
Features: facet=true & hl=true

TABLE I. QUERIES IN THE DATABASE

id	terms	filters	features
1	rnc50 OR uncorrelated	wt=json & rows=50 & start=0 & facet.field=RNC & facet.field=data_source & hl.fl=* & hl.snippets=3 & facet.field=Problem_text & fl=RNC,index_id,RBS	facet=true & hl=true
2	failure OR rnc80	wt=json & rows=50 & start=0 & facet.field=RNC & facet.field=data_source & hl.fl=* & hl.snippets=3 & facet.field=Problem_text & fl=RNC,index_id,RBS	facet=true & hl=true
3	failure	wt=json & rows=50 & start=0 & facet.field=RNC & facet.field=data_source & hl.fl=* & hl.snippets=3 & facet.field=Problem_text & fl=RNC,index_id,RBS	facet=true & hl=true
4	uncorrelated	wt=json & rows=50 & start=0 & facet.field=RNC & facet.field=data_source & hl.fl=* & hl.snippets=3 & facet.field=Problem_text & fl=RNC,index_id,RBS	facet=true & hl=true
5	rnc95 OR failure	wt=json & rows=50 & start=0 & facet.field=RNC & facet.field=data_source & hl.fl=* & hl.snippets=3 & facet.field=Problem_text & fl=RNC,index_id,RBS	facet=true & hl=true

TABLE II. RECOMMENDATION PROCESS

id	SLM_1_i	SLM_2_i	SLM_3_i	SLM_i	α_u_1	α_u_2	α_u_3	α_u_4	r	Final Score
1	-0.6667	-0.1581	1	0.1752	0	1	1	0.7333	2.1333	0.438
2	0	-0.1581	1	0.8418	0.8863	0	1	0.8667	2.7758	2.1047
3	0.5	-0.1581	1	1.3418	0.9516	0	0	0.3333	1.5946	3.3547
4	-0.5773	-0.1581	1	0.2645	0.9499	0	0	0.4	1.6599	0.6613
5	0	-0.1581	1	0.8418	0.9734	0	1	0.6667	2.6454	2.1047

Network management data has certain characteristics that affect its usefulness: Certain network events may only be valuable within a certain time range before going stale (e.g. last 5 minutes); Alarms may be sorted by severity; Search results may be grounded to make sense; and some keyword combinations may not have matching results.
In summary, the key advantages are:

- Reducing the complexity and time (interaction) needed for users to find final results and to improve search efficiency
- Reducing the use of system resources by avoiding unnecessary queries

IV. SYSTEM APPLICATION

The main goal of this search architecture is to assist the user to find and solve network problems with fewer interactions with the search interface. To demonstrate a prototype front-end search interface, we used third party web interface called HUE[3] from Cloudera. We developed a new standard search interface, and a new widget to optionally view and use recommended queries recommended by the recommender system. The search system can then be invoked with wither the user's query or the recommended query.

Data that is connected with the user and the queries are stored in a RDBMS database. We also have Ericsson network management data indexed in Apache Solr to test the system and see the results and possible recommendations. We choose Solr instead or an RDBMS because existing management data is not updated (read-only), rather new data is constantly inserted. In this scenario an indexing mechanism is more effective than a traditional database approach, thus allowing faster data retrieval.

The recommender module works as an independent plugin. If it is activated, it returns and presents recommendations depending on the query that the user performed. This is a pluggable and extensible approach. After every search, the user will be presented with an optional recommendation and different results, facets, charts and so on depends on what was chosen before. If the user clicks on a chart, a filter is applied, thus creating a new query to Solr. Therefore, the user will then have an updated recommendation.

As an example scenario: In a mobile telecoms network a Radio Network Controller node (RNC) is responsible for controlling and managing multiple base stations in a wireless network (UMTS). If a node (network element RNC75) has a problem, and assuming that previous troubleshooting the user usually finds that the problem is somehow liked to a problem in node RNC85, the system will over time learn and create an association between RNC75 and RNC85. This means that the next time that the user searches for RNC75 the system will recommend examining data from RNC75 and RNC85 to troubleshooting the problem.

Another possible scenario is when an OSS engineer is searching for alarms. If she usually sorts the results by priority, next time the system will recommend this sort criterion.

[3] http://gethue.com/

The main advantage with this system is that the user can find the optimal result with fewer iterations. In the previous system the user needed to look for a specific RNC or fault. Then, in the result section, he needed to examine partial results and check if there were any other RNCs correlated with this issue. In addition the user would have to apply the filters that she wants. With PRISENIT, such interactions are learned and next time offered to the user. The only thing that the user has to do is to select the recommended search.

V. EXPERIMENTAL EVALUATION

For the evaluation of PRISENIT, we used data from an alarm list, performance counter values, logs and performance event logs derived from a text network in an Ericsson test lab.

A. *Performance*

For all evaluations we used a Ubuntu 12.04 (precise) 64-bit system with Intel® Core™ i5-2540M CPU @ 2.6GHz x 4 and 4GB of RAM.

Fig. 3 shows the time that the system needed to create a new suggestion for the user, depending on how many queries are analyzed. The experiments were conducted multiple times and averaged in order to compensate for other random factors that may influence the response times (such us usage of the system). In this case we can appreciate that the increase of response time of the system is somewhat linearly related to the number of existing queries stored in the database. According to [16], the tolerable waiting time it is around 2 seconds, so the response time of system remains acceptable. In real network management deployments, there should be little need to examine queries more than a few days old, so a limitation to examine only several thousand existing relevant queries is reasonable.

In our deployments the terms are often short and tend to be composed by one or two keywords. In this system, the filter part of the queries often has more data. For example, in our system there are many facets to show results grouped by specific fields with specific filed filters, such as sorting, limits, type and so on.

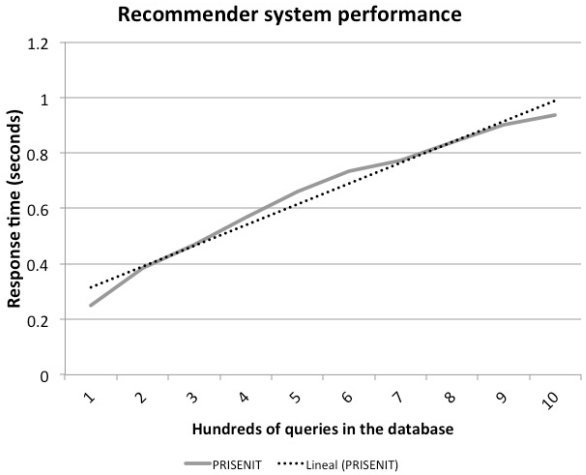

Figure 3. Online time necessary to generate the suggestion varying the number of analyzed queries.

B. *Usability and usefulness*

We conduct our usability assessment using a questionnaire. In this case, usability focuses on the effectiveness of the system for network intelligence and troubleshooting from the point of view of the end user (network engineers). We select the System Usability Scale (SUS) [17] because it is technology agnostic, making it flexible enough to asses a wide range of interface technologies, and allows quick and easy assessment of the usability of a given system.

In [18] it is stated that a sample of 12 users is sufficient to provide correct findings (i.e. the same score as a larger sample) 90–100% of the time and a sample of only 10 users, 75–80% of the time. Moreover, the original SUS instrument is composed of 10 statements that are scored on a 5-point scale of strength of agreement, among which odd questions are positive statements and the rest are negative. Final scores for the SUS can range from 0 to 100, where higher scores indicate better usability. Table 3 shows our SUS results.

TABLE III. SUS SCORING INSTRUMENT

Users / Queries	1	2	3	4	5	6	7	8	9	10	11	12	*Average*
q1	4	5	4	4	3	5	5	5	3	4	5	4	**4.25**
q2	2	1	2	2	2	2	1	1	1	2	1	3	**1.67**
q3	3	5	3	3	4	5	4	5	5	4	4	5	**4.17**
q4	1	1	3	3	4	2	2	2	4	1	1	2	**2.17**
q5	3	5	4	5	5	5	5	5	5	5	4	5	**4.67**
q6	1	1	1	1	2	1	1	1	1	1	1	1	**1.08**
q7	5	4	4	4	5	5	5	5	4	4	4	4	**4.42**
q8	2	1	2	2	2	1	1	1	2	1	2	1	**1.5**
q9	4	5	3	4	4	5	4	5	3	5	5	5	**4.33**
q10	1	1	2	2	3	2	2	3	2	1	2	1	**1.83**
Scores	**80**	**97.5**	**70**	**75**	**70**	**92.5**	**90**	**92.5**	**75**	**90**	**87.5**	**87.5**	***83.96***

The survey was completed by network engineers. Evaluating our system, table 3 shows an average score of 83.96. When converted to percentiles [19] this score means that we can get a grade A for our system. Using the scale used in [20] the grade of our system would be grade B. Moreover, the system would be described as "good" and would be considered acceptable. Besides, some of the participant (OSS engineers) point out that this system was much better that what they have at this moment. They said that it is easier to use, more intuitive and the recommender part can help them for troubleshooting.

VI. Conclusion and Future Work

The main purpose of this work is to develop a system to improve search efficiency for network monitoring and troubleshooting and to make user experience easier and more efficient by reducing costly human interaction with the system.

While this paper focuses on finding a comprehensive and robust approach to calculate the similarity between user interactions, such an approach will contribute significantly in a wide variety of telecom network management use cases apart from alarm filtering and correlation. The system is capable of dynamically creating associations between attributes. Furthermore, the system updates these associations as the user necessities are changing.

Moreover, this system is extendable and pluggable. The actual system can work without the recommender system. Hence, the user can choose if he wants to see a recommendation or not.

From our evaluation we saw that the overhead of the system increases somewhat linearly with the number of previous searches stored. Therefore it will be necessary to create a mechanism to periodically remove searches that are found not to be reusable or widely relevant.

It would also be interesting to compare this approach to one where recommendations are generated for terms, filters and features separately, rather than generating an entire query. This could be achieved by creating separate recommendations with single action-based weightings for terms, filters and features. On the other hand, another possible approach would be to use different weightings to increase the importance of term similarity rather than filter/feature similarities.

It is also planned to enhance this approach with a prediction mechanism to proactively present recommended searches before or as interesting faults or events occur in the network.

References

[1] Wang, M., Handurukande, S. B., and Nassar, M., "RPig: A scalable framework for machine learning and advanced statistical functionalities," Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on, pp. 293-300, December 2012.

[2] Ben-Yitzhak, O., Golbandi, N., Har'El, N., Lempel, R., Neumann, A., Ofek-Koifman, S., Sheinwald, D., Shekita, E., Sznajder, B., and Yogev, S., "Beyond basic faceted search," *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pp. 33-44, 2008.

[3] Azad, H. K. and Abhishek, K., "Semantic-Synaptic Web Mining: A Novel Model for Improving the Web Mining," *Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on*, pp. 454-457, April 2014.

[4] Sampath, P., Ramesh, C., Kalaiyarasi, T., Banu, S. S., and Selvan, G. A., "An efficient weighted rule mining for web logs using systolic tree," *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on*, pp. 432-436, March 2012.

[5] Nasraoui, O., Soliman, M., Saka, E., Badia, A., and Germain, R., "A Web Usage Mining Framework for Mining Evolving User Profiles in Dynamic Web Sites," *Knowledge and Data Engineering, IEEE Transactions on*, vol.20, no.2, pp. 202-215, Feb 2008.

[6] Yang, Y., Aufaure, M., and Claramunt, C., "Towards a DL-Based Semantic User Model for Web Personalization," *Autonomic and Autonomous Systems, 2007. ICAS07. Third International Conference on*, pp. 61-61, June 2007.

[7] Zhou, X., Wu, S.-T., Li, Y., Xu, Y., Lau, R. Y. K., and Bruza, P. D., "Utilizing Search Intent in Topic Ontology-Based User Profile for Web Mining," *Web Intelligence, 2006. WI 2006. IEEE/WIC/ACM International Conference on*, pp. 558-564, December 2006.

[8] Zhuhadar, L., Nasraoui, O., Wyatt, R., and Romero, E., "Multi-model Ontology-Based Hybrid Recommender System in E-learning Domain," *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT '09. IEEE/WIC/ACM International Joint Conferences on*, vol.3, pp. 91-95, September 2009.

[9] Zhou, B., Hui, S. C., and Chang, K., "An intelligent recommender system using sequential Web access patterns," *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, vol.1, pp. 393-398, Dec 2004.

[10] Resnick, P. and Varian, H. R., "Recommender systems," *Communications of the ACM*, vol.40, no.3, pp. 56-58, 1997.

[11] Rich, E., "User modeling via stereotypes*," *Cognitive science*, vol.3, no.4, pp. 329-354, 1979.

[12] Uddin, M., Stadler, R., and Clemm, A., "A query language for network search," *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*,, pp. 109-117, May 2013.

[13] Keeney, J., van der Meer, S., and Hogan, G., "A recommender-system for telecommunications network management actions," *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pp. 760-763, May 2013.

[14] Tapucu, D., Kasap, S., and Tekbacak, F., "Performance Comparison of Combined Collaborative Filtering Algorithms for Recommender Systems," *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual*, pp. 284-289, July 2012.

[15] Bobadilla, J., Serradilla, F., and Gutierrez, A., "Recommender systems: Improving collaborative filtering results," *ICT and Knowledge Engineering, 2009 7th International Conference on*, pp. 100-106, Dec 2009.

[16] Nah, F. F.-H., "A study on tolerable waiting time: how long are Web users willing to wait?.," *Behaviour & Information Technology*, vol.23, no.3, pp. 153 - 163, 2004.

[17] Brooke, J., "SUS-A quick and dirty usability scale," *Usability evaluation in industry*, vol.189, pp. 194, 1996.

[18] Tullis, T. S. and Stetson, J. N., "A comparison of questionnaires for assessing website usability," *Usability Professional Association Conference*, pp. 1-12, 2004.

[19] Bangor, A., Kortum, P., and Miller, J., "Determining what individual SUS scores mean: Adding an adjective rating scale," *Journal of usability studies*, vol.4, no.3, pp. 114-123, 2009.

[20] Sauro, J., *A practical guide to the system usability scale: Background, benchmarks & best practices*, Measuring Usability LLC, 2011.